LAB 2 - Node Exporter

Création d'utilisateurs de service

Créez les utilisateurs et utilisez les options `--no-create-home` et `--shell /bin/false` afin que ces utilisateurs ne puissent pas se connecter au serveur.

```
sudo useradd --no-create-home --shell /bin/false node_exporter
```

Téléchargement de Node Exporter

Pour étendre Prometheus au-delà des métriques sur lui-même uniquement, nous allons installer un exportateur supplémentaire appelé **Node Exporter**. Node Exporter fournit des informations détaillées sur le système, notamment l'utilisation du processeur, du disque et de la mémoire.

Tout d'abord, téléchargez la version stable actuelle de Node Exporter dans votre répertoire personnel. Vous pouvez trouver les derniers binaires ainsi que leurs sommes de contrôle sur la page de téléchargement de Prometheus : https://prometheus.io/download/#node_exporter

```
cd ~
curl -LO
https://github.com/prometheus/node_exporter/releases/download/v1.1.0/node_exporter-
1.1.0.linux-amd64.tar.gz
```

Utilisez la sha256sum commande pour générer une somme de contrôle du fichier téléchargé:

```
sha256sum node_exporter-1.1.0.linux-amd64.tar.gz
```

Vérifiez l'intégrité du fichier téléchargé en comparant sa somme de contrôle avec celle de la page de

Si les sommes de contrôle ne correspondent pas, supprimez le fichier téléchargé et répétez les étapes précédentes.

Maintenant, décompressez l'archive téléchargée.

```
tar xvf node_exporter-1.1.0.linux-amd64.tar.gz
```

Cela créera un répertoire appelé node_exporter-1.1.0.linux-amd64 contenant un fichier binaire nommé node_exporter, une licence et un avis.

Copiez le binaire dans le répertoire /usr/local/bin et définissez la propriété de l'utilisateur et du groupe à l' utilisateur **node_exporter** que vous avez créé à l'étape 1.

```
sudo cp node_exporter-1.1.0.linux-amd64/node_exporter /usr/local/bin
sudo chown node_exporter:node_exporter /usr/local/bin/node_exporter
```

Enfin, supprimez les fichiers restants de votre répertoire personnel car ils ne sont plus nécessaires.

```
rm -rf node_exporter-1.1.0.linux-amd64.tar.gz node_exporter-1.0.1.linux-amd64
```

Maintenant que vous avez installé Node Exporter, testons-le en l'exécutant avant de créer un fichier de service pour lui afin qu'il se lance au démarrage.

Exécution de Node Exporter

Les étapes pour exécuter Node Exporter sont similaires à celles pour exécuter Prometheus lui-même. Commencez par créer le fichier de service Systemd pour Node Exporter.

```
sudo nano /etc/systemd/system/node_exporter.service
```

Ce fichier de service indique à votre système d'exécuter Node Exporter en tant qu'utilisateur **node_exporter** avec l'ensemble de collecteurs par défaut activé.

Copiez le contenu suivant dans le fichier de service:

Fichier de service Node Exporter - **/etc/systemd/system/node_exporter.service**

```
[Unit]
Description=Node Exporter
Wants=network-online.target
After=network-online.target

[Service]
User=node_exporter
Group=node_exporter
Type=simple
ExecStart=/usr/local/bin/node_exporter

[Install]
WantedBy=multi-user.target
```

Les collecteurs définissent les métriques que Node Exporter générera. Vous pouvez voir la liste complète des collecteurs de Node Exporter - y compris ceux qui sont activés par défaut et ceux qui sont obsolètes - dans le fichier README de Node Exporter.

Si vous devez remplacer la liste par défaut des collecteurs, vous pouvez utiliser l'indicateur --collectors (pour la liste des collecteurs visitez)

Enfin, rechargez systemd pour utiliser le service nouvellement créé.

```
sudo systemctl daemon-reload
```

Vous pouvez maintenant exécuter Node Exporter à l'aide de la commande suivante:

```
sudo systemctl start node_exporter
```

Vérifiez que Node Exporter fonctionne correctement avec la commande `status`.

```
sudo systemctl status node_exporter
```

Comme auparavant, cette sortie vous indique l'état de Node Exporter, l'identificateur de processus principal (PID), l'utilisation de la mémoire, etc.

Si l'état du service n'est pas active, suivez les messages à l'écran et retracez les étapes précédentes pour résoudre le problème avant de continuer.

Enfin, activez Node Exporter pour démarrer au démarrage.

```
sudo systemctl enable node_exporter
```

Avec Node Exporter entièrement configuré et fonctionnant comme prévu, nous dirons à Prometheus de commencer à récupérer les nouvelles métriques.

Configuration de Prometheus pour lire Node Exporter

Étant donné que Prometheus ne suit que les exportateurs définis dans la partie scrape_configs de son fichier de configuration, nous devons ajouter une entrée pour Node Exporter, tout comme nous l'avons fait pour Prometheus lui-même.

Ouvrez le fichier de configuration.

```
sudo nano /etc/prometheus/prometheus.yml
```

À la fin du bloc `scrape_configs`, ajoutez une nouvelle entrée appelée node_exporter.
Fichier de configuration Prometheus partie 1 - /etc/prometheus/prometheus.yml

```
...

- job_name: 'node_exporter'
  scrape_interval: 5s
  static_configs:
  - targets: ['localhost:9100']
```

Parce que cet exportateur est en cours d'exécution aussi sur le même serveur que Prometheus, nous pouvons utiliser localhost au lieu d'une adresse IP à nouveau avec le port par défaut du Node Exporter, 9100.

L'ensemble de votre fichier de configuration devrait ressembler à ceci:

Fichier de configuration Prometheus - /etc/prometheus/prometheus.yml

```
global:
  scrape_interval: 15s

scrape_configs:
- job_name: 'prometheus'
  scrape_interval: 5s
  static_configs:
  - targets: ['localhost:9090']
- job_name: 'node_exporter'
  scrape_interval: 5s
  static_configs:
  - targets: ['localhost:9100']
```

Enregistrez le fichier et quittez votre éditeur de texte lorsque vous êtes prêt à continuer.

Enfin, redémarrez Prometheus pour appliquer les modifications.

```
sudo systemctl restart prometheus
```

Encore une fois, vérifiez que tout fonctionne correctement avec la commande status.

```
sudo systemctl status prometheus
```

Si l'état du service n'est pas défini sur active, suivez les instructions à l'écran et retracez vos étapes précédentes avant de continuer.

Nous avons maintenant Prometheus et Node Exporter installés, configurés et en cours d'exécution.

Working with exporters

Node Exporter

Le Node Exporter est probablement l'un des premiers exportateurs que vous utiliserez. Il expose des métriques au niveau de la machine, en grande partie à partir du noyau de votre système d'exploitation, telles que le processeur, la mémoire, l'espace disque, les E/S disque, la bande passante réseau et la température de la carte mère. L'exportateur de nœuds est utilisé avec les systèmes Unix; Les utilisateurs de Windows doivent utiliser wmi_exporter à la place.

Node Exporter est uniquement destiné à surveiller la machine elle-même, pas les processus ou services individuels sur celle-ci. Dans l'architecture Prometheus, chacun de vos services exposera ses propres métriques, en utilisant un exportateur si nécessaire, qui est ensuite directement récupéré par Prometheus. Cela vous évite de vous retrouver avec un goulot d'étranglement opérationnel ou de performance, et vous permet de penser davantage en termes de services dynamiques que de machines.

Dans le cas de Linux, des milliers de métriques sont proposées. Certains sont bien documentés et compris, comme l'utilisation du processeur; d'autres, comme l'utilisation de la mémoire, ont varié d'une version du noyau à l'autre car l'implémentation a changé. Vous trouverez même des métriques complètement non documentées, où vous devrez lire le code source du noyau pour essayer de comprendre ce qu'elles font. Node Exporter est conçu pour être exécuté en tant qu'utilisateur non root et doit être exécuté directement sur la machine de la même manière que vous exécutez un démon système tel que sshd ou cron.

Contrairement à la plupart des autres exportateurs, en raison de la grande variété de mesures disponibles à partir des systèmes d'exploitation, le Node Exporter vous permet de configurer les catégories de mesures qu'il récupère. Vous pouvez le faire avec des indicateurs de ligne de commande tels que **--collector.wifi**, qui activerait le collecteur WiFi, et **--no-collector.wifi**, qui le désactiverait. Il y a des valeurs par défaut raisonnables, donc ce n'est pas quelque chose dont vous devriez vous soucier au début.

CPU Collector

La métrique principale du collecteur de CPU est **node_cpu_seconds_total**, qui est un compteur indiquant le temps que chaque CPU a passé dans chaque mode. Les libellés sont cpu et mode.

```
# HELP node_cpu_seconds_total Seconds the cpus spent in each mode.
# TYPE node_cpu_seconds_total counter
node_cpu_seconds_total{cpu="0",mode="idle"}           48649.88
node_cpu_seconds_total{cpu="0",mode="iowait"}         169.99
node_cpu_seconds_total{cpu="0",mode="irq"}            0
node_cpu_seconds_total{cpu="0",mode="nice"}           57.5
node_cpu_seconds_total{cpu="0",mode="softirq"}        8.05
node_cpu_seconds_total{cpu="0",mode="steal"}          0
node_cpu_seconds_total{cpu="0",mode="system"}         1058.32
node_cpu_seconds_total{cpu="0",mode="user"}           4234.94
node_cpu_seconds_total{cpu="1",mode="idle"}           9413.55
node_cpu_seconds_total{cpu="1",mode="iowait"}         57.41
node_cpu_seconds_total{cpu="1",mode="irq"}            0
node_cpu_seconds_total{cpu="1",mode="nice"}            46.55
node_cpu_seconds_total{cpu="1",mode="softirq"}         7.58
node_cpu_seconds_total{cpu="1",mode="system"}         1034.82
node_cpu_seconds_total{cpu="1",mode="user"}           4285.06
```

Pour chaque CPU, les modes augmente globalement d'une seconde, par seconde. Cela vous permet de calculer la proportion de temps d'inactivité sur tous les processeurs à l'aide de l'expression PromQL:

```
avg without(cpu, mode)(rate(node_cpu_seconds_total{mode="idle"}[1m]))
```

Cela fonctionne car il calcule le temps d'inactivité par seconde par CPU, puis fait la moyenne de tous les CPU de la machine. Vous pouvez généraliser cela pour calculer la proportion de temps passé dans chaque mode pour une machine en utilisant:

```
avg without(cpu)(rate(node_cpu_seconds_total[1m]))
```

Filesystem Collector

Le collecteur de système de fichiers collecte sans surprise des métriques sur vos systèmes de fichiers montés, comme vous le feriez avec la commande df.
Les indicateurs **`--collector.filesystem.ignored-mount-points`** et **`--collector.filesystem.ignored-fs-types`** permettent de restreindre les systèmes de fichiers inclus (les valeurs par défaut excluent divers pseudo systèmes de fichiers). Étant donné que Node Exporter ne fonctionnera pas en tant que root, vous devrez vous assurer que les autorisations de fichier lui permettent d'utiliser l'appel système `statfs` sur les points de montage qui vous intéressent. Toutes les métriques de ce collecteur sont préfixées par **`node_filesystem_`** et ont des étiquettes de **`device`**, **`fstype`**, et **`mountpoint`**.

```
# HELP node_filesystem_size_bytes Filesystem size in bytes.
# TYPE node_filesystem_size_bytes gauge
node_filesystem_size_bytes{device="/dev/sda1",fstype="ext4",mountpoint="/"}
```

Les métriques du système de fichiers sont largement évidentes. La seule subtilité dont vous devez être conscient est la différence entre **node_filesystem_avail_bytes** et **node_filesystem_free_bytes**. Sur les systèmes de fichiers Unix, l'espace est réservé à l'utilisateur root, afin qu'il puisse encore continuer à utiliser le système lorsque les utilisateurs remplissent tout l'espace disponible.
node_filesystem_avail_bytes est l'espace disponible pour les utilisateurs, et lorsque vous essayez de calculer l'espace disque utilisé, vous devez utiliser en conséquence:

```
node_filesystem_avail_bytes / node_filesystem_size_bytes
```

Diskstats Collector

Le collecteur diskstats expose les métriques d'E/S disque à partir de /proc/diskstats. Par défaut, l'indicateur **`--collector.diskstats.ignored-devices`** tente d'exclure les éléments qui ne sont pas de vrais disques, tels que les partitions et les périphériques de loopback:

```
# HELP node_disk_io_now The number of I/Os currently in progress.
# TYPE node_disk_io_now gauge
node_disk_io_now{device="sda"}
```

Toutes les métriques ont une étiquette **device**, et presque toutes sont des compteurs, comme suit:

```
node_disk_io_now  : The number of I/Os in progress.
node_disk_io_time_seconds_total : Incremented when I/O is in progress.
node_disk_read_bytes_total : Bytes read by I/Os.
node_disk_read_time_seconds_total : The time taken by read I/Os.
node_disk_reads_completed_total : The number of complete I/Os.
node_disk_written_bytes_total : Bytes written by I/Os.
node_disk_write_time_seconds_total : The time taken by write I/Os.
node_disk_writes_completed_total : The number of complete write I/Os.
```

Celles-ci signifient principalement ce que vous pensez, mais jetez un œil à la documentation du kernel pour plus de détails. Vous pouvez utiliser node_disk_io_time_seconds_total pour calculer l'utilisation du disque E/S, comme le montrerait `iostat -x`:

```
rate(node_disk_io_time_seconds_total[1m])
```

Vous pouvez calculer le temps moyen d'une lecture de E/S avec:

```
rate(node_disk_read_time_seconds_total[1m]) /  rate(node_disk_reads_completed_total[1m])
```

Netdev Collector

Le **Netdev** Collector expose des métriques sur vos périphériques réseau avec le préfixe **`node_network_`** et une étiquette **`device`**.

```
# HELP node_network_receive_bytes_total Network device statistic receive_bytes.
# TYPE node_network_receive_bytes_total counter
node_network_receive_bytes_total{device="lo"}              8.3213967e+07
node_network_receive_bytes_total{device="enp03"}           7.0854462e+07
```

node_network_receive_bytes_total et **node_network_transmit_bytes_total** sont les principales métriques qui vous intéressent car vous pouvez calculer la bande passante du réseau avec elles.

```
rate(node_network_receive_bytes_total[1m])
```

Vous pouvez également être intéressé par node_network_receive_packets_total et node_network_transmit_packets_total, qui suivent respectivement les paquets entrants et sortants.

Meminfo Collector

Le collecteur **meminfo** a toute votre métrique standard liée à la mémoire avec un préfixe **node_memory_**. Tout cela vient de votre /proc/meminfo, et c'est le premier collecteur où la sémantique devient un peu confuse. Le collecteur convertit les kilo-octets en octets préférés, mais au-delà de cela, il vous appartient d'en savoir suffisamment grâce à la documentation et à l'expérience des internes Linux pour comprendre ce que signifient ces métriques:

```
# HELP node_memory_MemTotal_bytes Memory information field MemTotal.
# TYPE node_memory_MemTotal_bytes gauge
node_memory_MemTotal_bytes              8.269582336e+09
```

Par exemple, **node_memory_MemTotal_bytes** est la quantité totale de mémoire physique dans la machine . Mais notez qu'il n'y a pas de métrique de mémoire utilisée, donc vous devez en quelque sorte la calculer et aussi combien de mémoire n'est pas utilisée à partir d'autres métriques. **node_memory_MemFree_bytes** est la quantité de mémoire qui n'est utilisée par rien, mais cela ne veut pas dire que c'est toute la mémoire dont vous disposez. En théorie, votre cache de swap (**node_memory_Cached_bytes**) peut être récupéré, tout comme vos tampons d'écriture (**node_memory_Buffers_bytes**), mais cela pourrait nuire aux performances de certaines applications. En outre, il existe diverses autres structures de noyau utilisant de la mémoire telles que slab est page tables. **node_memory_MemAvailable** est une heuristique du noyau pour la quantité de mémoire réellement disponible, mais n'a été ajoutée que dans la version 3.14 de Linux. Si vous exécutez un noyau suffisamment neuf, il s'agit d'une métrique que vous pouvez utiliser pour détecter l'épuisement de la mémoire.